IJCSE International Journal of Computer Sciences and Engineering Open Access

Research Paper Vol.-7, Issue-2, Feb 2019 E-ISSN: 2347-2693

Improved Software Cost Estimation Model Using Cost Driver Reduction Based on Water Cycle Algorithm

Zahid Hussain Wani[1], Kaiser Javeed Giri[2,*], Rumaan Bashir[3]

[1,2,3]Department of Computer Sciences, Islamic University of Science & Technology, J&K, India

[*]*Corresponding Author: kaiserjaveed@gmail.com, Tel.: +91-9419167696*

DOI: https://doi.org/10.26438/ijcse/v7i2.6472 | Available online at: www.ijcseonline.org

Accepted: 22/Feb/2019, Published: 28/Feb/2019

***Abstract* —** Software cost estimation considered to be the critical, is equally vital tasks in software project management. In a highly challenging environment, software project managers are always in a need of robust estimation models inorder to predict the cost of upcoming software development projects accurately. Software cost estimation is the prediction of development effort and calendar time required to develop a software project. It is considered to be the key task as accurate estimation of any software not only accurately estimates development effort, cost, time and growth of a software development project but also yields delivery exactness and correctness vis a viz return an organization in a better schedule of its futuristic software projects. In this paper, software cost estimation is done by proposing a cost driver selection model which is based on an optimization technique called as water cycle algorithm. The proposed cost driver selection model selects only relevant set of cost drivers as an input to estimation process and ignores the very irrelevant cost drivers. In step second, these relevant set of cost drivers originating from step first are assigned to an Artificial Neural Network as its input for the purpose of getting the accurate estimation of software development project cost that needs to be developed. For evaluation purposes, Magnitude of Relative Error, Mean of Magnitude of Relative Error and Median of Magnitude of Relative Error are used as three performance measures to simply weigh the obtained quality of estimation as accuracy. The obtained results were compared with the results of a benchmark study of COCOMO model and another artificial neural network based model. From the comparative result, it becomes evident that the proposed model outperforms the rest of the two existing models.

***Keywords*—** Artificial Neural Network, Cost Driver Reduction, Software Cost Estimation, Water Cycle Algorithm

I. INTRODUCTION

Software cost estimation is defined as the estimation of findings of cost and time for any software project. It is never an exact science and the reason being involvement of large number of cost drivers based on which the estimation is done. Furthermore, any process that involves a significant involvement of human factor can never be exact because humans are far too complex to be entirely expected. Software cost estimation is one of the most vital aspects of software development process. An inappropriate estimation of software development costs of any software product always make the project manager incapable of analyzing time and effort required for the project, thus leading to its over budgeting and time deadline extensions. Thus, software cost estimation is very significant tool as it affects both planning and budgeting of a project and equally challenging. From earlier discussion, it becomes evident that the success of every software development project is predominantly based on the accuracy of its cost estimation. An effective accuracy in estimation of software cost causes project managers to keep track of every activity of ongoing project and at the same time let the organization to gain an insight of successful inception of futuristic projects by making efficient utilization of all of its resources including human resources like analyzers, designers, programmers & other alike work forces and some other Non-human resources. Thus accurate software cost estimation besides serving in successful completion of the currently running software development projects, also results in the overall development of the software development organization. Thus, predictions for new projects should be as close to the actual cost as possible inorder to avoid sufferings in terms of resources and standing against the deadline timings.

The rest of the paper is organized as follows: In Section II, a detailed description of various existing techniques in the field of software cost estimation is given, followed by a brief discussion about water cycle model that is used in attribute reduction as an optimization algorithm is presented in section III. Proposed model of cost drivers reduction and artificial neural network will be presented in section IV. Section V presents the details of datasets and the evaluation criteria used in the study. Section VI reports to the experimentation results of SCE generated using the proposed model, followed by comparative analysis of achieved results against the results of two existing techniques to be presented in the same section. Section VII concludes the proposed work and its future scope.

II. Related Work

Several software cost estimation methods have been proposed in past to make accurate predictions that help to complete within predicted budget and on time [1][2][3][4][5][6][7][8][9]. These models include expert-based, analogy-based and regression-based estimation methods [7][8] [9] [10][11][12] [13][14]. Expert judgment based estimation method is considered to be less accurate [7] as this kind of estimation method is solely dictated by the expert to be consulted and thereby gets easily biased based on the expert's experience [14]. Analogy-based software cost estimation model solves the estimation problem by consulting project information from early completed projects and using the same into the current one [15]. Thus in this estimation model, past data of historical software development projects becomes the base for estimation of software cost and time of the project to be developed. In general analogy-based estimation models, a feature vector is generated and accordingly used to calculate distance between a history project and the new one. This feature vector more sprecifically in case of quantitative data with no missing values is used in the estimation model [1][16] [5][6][17]. Many approaches propose selection methods of features [18], clorresponding assignment of weights to these features, normally either considering Euclidean distance [1][19][9], or fuzzy logic [20], or rough set analysis [6][8] or genetic algorithms [5]. Contrary to above two types of models, Regression-based cost estimation models use statistical algorithms inorder to convert features as independent variables into cost as a dependent variable, with minimum possible errors [13].

Influenced from the use of machine learning techniques and their impact in solving problems of different domains [21] of challenging prediction problems like From recent past, a good number of alternative modeling techniques have also been proposed. These include artificial neural networks [22] also providing pathways to more sophisticated convolution neural networks dominantly and accurately meant for many feature selection in-case of image processing and optimization problems [23]. The others include regression trees and rule induction models. Authors in [24] offered several other different predictive model-building techniques such as robust statistical procedures, various forms of neural network models, fuzzy logic, case-based reasoning and regression trees. Predictive modelling using artficial neural networks is also presented many researchers like [25]. Authors in [26] presented another innovative technique for SCE based on fuzzy identification. On comparison of his technique with three Basic, Intermediate, and Detailed COCOMO estimation models resulted in significant achievement of better estimates. There are many other researchers, who presented their research studies based on neural networks approach for the purpose of estimating software development effort[27] [28] [29] [30] [31] [32] [33] [34]. Most of their studies have fixed more attention on the accuracy of other cost estimation techniques like COCOMO and Function Point Analysis.

III. WATER CYCLE ALGORITHM

Studying form the natural world and observing the natural process of water cycle as how streams and rivers flow down toward a sea, Eskandar et al. proposed the water cycle algorithm [35]. As water flows down from higher place to lower one, a river or a stream is created. As such, most rivers are formed at the top of mountains where the melting of snow occurs. In turn, the rivers constantly flow down and along this journey they are fed with water from rainfall and from other streams before they subsequently end up in the sea. A simple diagram depicting part of this water cycle is given in Figure 1 as:

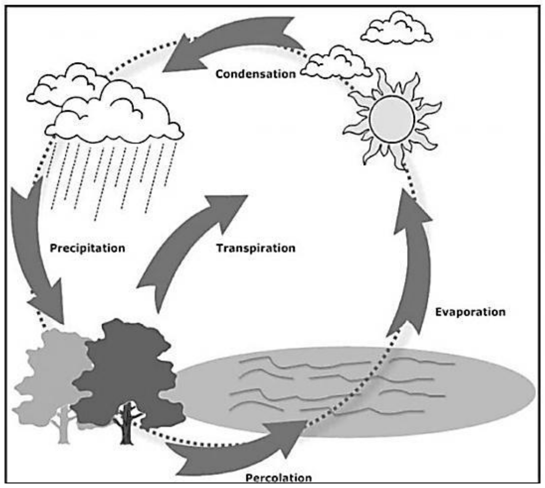

Figure 1. A Simplified diagram of the hydrologic cycle (water cycle) by Eskandar et al. [35]

The water in lakes and rivers start to evaporate. Moreover, during the process of photosynthesis plants either give off or transpire water. Then, the water that is evaporated or transpired goes up into the atmosphere and leads to the formation of clouds that condense in the colder air above. Thus the water is circulated through precipitation and the formation of rain back to the earth again. This process is known as the hydrologic or water cycle [36]. In our natural world, most of the water that comes from the melting of snow or from rainfall seeps into the permeable layer of rock or soil underground and is stored there in large amounts. This aquifer is sometimes referred to as groundwater for more clarification (see percolation arrow in Fig. 1. That water in the aquifer flows in a downward direction underground in the same way that it flows on the surface of the ground. The underground water could be emptied into a lake, swamp or stream. More clouds are formed through the evaporation of water from streams and rivers, together with transpiration from trees and other vegetation, thus causing more rain to fall, and so the cycle goes on.

IV. The Proposed Attribute Reduction Based Software Cost Estimation Model

Step 1: Proposed Cost Driver Reduction Procedure

In this study, binary representation is adopted to represent each solution. In this representation, a candidate solution is denoted as a one-dimensional array with a fixed size. The size of the array is equal to the number of attributes (Z) in a given problem dataset. Each cell of the array is set to value "1" or "0" depends whether the attribute is considered or not. If considered, it is set to "1" else set to "0" if not selected. An initial solution can be generated by random assignment of cells with "1" or "0". An example of the adopted representation is shown in Figure 1, where the number of attributes is Z=17 and each cell assigned to "0" or "1". This example indicates that attributes number 1, 3, 5, 9, 12, 14 and 17 have been selected, while other is not.

Table 1. Solution Representation (Length, Z=17)

1	0	1	0	1	0	0	0	1	0	0	1	0	1	0	0	1

In order to solve an optimization problem using population based metaheuristic methods, it is necessary that the values of problem variables be formed as an array. In GA and PSO terminologies such array is called ''Chromosome'' and ''Particle Position'', respectively. Accordingly, in the proposed method it is called ''Raindrop'' for a single solution. In an Nvar dimensional optimization problem, a raindrop is an array of *1 x Nvar*. This array is defined as follows:

$$Raindrop = [x_1, x_2, x_3 \ldots\ldots x_N] \qquad (2)$$

To start the optimization algorithm, a candidate representing a matrix of raindrops of size N_{pop} x N_{var} is generated (i.e. population of raindrops). Hence, the matrix X which is generated randomly is given as (rows and column are the number of population and the number of design variables, respectively):

$$Population\ of\ Raindrops = \begin{bmatrix} Raindrop\ 1 \\ Raindrop\ 2 \\ Raindrop\ 3 \\ \vdots \\ Raindrop\ n \end{bmatrix} \qquad (3)$$

$$= \begin{bmatrix} x_1^1 & x_2^1 & x_3^1 & . & . & x_{N_{var}}^1 \\ x_1^2 & x_2^2 & x_3^2 & . & . & x_{N_{var}}^2 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ x_1^{N_{pop}} & x_2^{N_{pop}} & x_3^{N_{pop}} & . & . & x_{N_{var}}^{N_{pop}} \end{bmatrix} \qquad (4)$$

Each of the decision variable values ($X_1, X_2, X_3 \ldots X_{Nvar}$) can be represented as floating point number (real values) or as a predefined set for continuous and discrete problems, respectively. The cost of a raindrop is obtained by the evaluation of cost function (C) given as:

$$C_i = Cost_i = f\left(x_1^i, x_2^i, \ldots\ldots x_{N_{var}}^i\right)$$
$$i = 1,2,3, \ldots\ldots, N_{Pop} \qquad (5)$$

where 'N_{pop}' and 'N_{vars}' are the number of raindrops (initial population) and the number of design variables, respectively. For the first step, 'N_{pop}' raindrops are created. A number of 'N_{sr}' from the best individuals (minimum values) are selected as sea and rivers. The raindrop which has the minimum value among others is considered as a sea. In fact, 'N_{sr}'' is the summation of Number of Rivers (which is a user parameter) and a single sea as given in Eq. (6). The rest of the population (raindrops form the streams which flow to the rivers or may directly flow to the sea) is calculated using Eq. (7).

$$N_{sr} = Number\ of\ Rivers + \underbrace{1}_{Sea} \qquad (6)$$

$$N_{Raindrops} = N_{pop} - N_{sr} \qquad (7)$$

In order to designate/assign raindrops to the rivers and sea depending on the intensity of the flow, the following equation is given:

$$N_{sn} = round\left\{\frac{Cost_n}{\sum_{i=1}^{N_{sr}} Cost_i} \times N_{Raindrops}\right\},$$
$$n = 1,2, \ldots\ldots, N_{sr} \qquad (8)$$

where 'NS_n' is the number of streams which flow to the specific rivers or sea.

As already mentioned above, the streams are created from the raindrops and join each other to form new rivers. Some of the streams may also flow directly to the sea. All rivers and streams end up in sea (best optimal point). Figure 2 shows the schematic view of stream's flow towards a specific river.

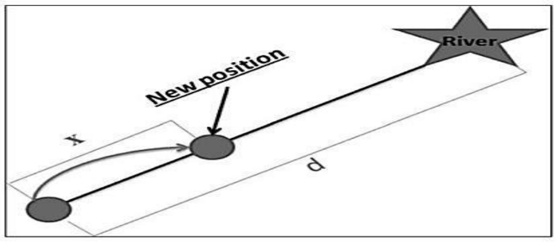

Figure 2. Schematic view of flow of a stream to a particular river (The river and stream are represented by the star and circle, respectively) by Eskandar et al. [35].

As illustrated in Fig. 2, a stream flows to the river along the connecting line between them using a randomly chosen distance given as follow:

$$X \in (0, C \times d), \quad C > 1 \tag{9}$$

where C is a value between 1 and 2 (near to 2). The best value for '*C*' may be chosen as 2. The current distance between stream and river is represented as '*d*'. The value of '*X*' in above Eq. (9) corresponds to a distributed random number (uniformly or may be any appropriate distribution) between 0 and (*C* × *d*). The value of '*C*' being greater than one enables streams to flow in different directions towards the rivers.

This concept may also be used in flowing rivers to the sea. Therefore, the new position for streams and rivers may be given as:

$$X_{Stream}^{i+1} = X_{Stream}^{i} + rand \times C \times \left(X_{River}^{i} - X_{Stream}^{i}\right) \tag{10}$$

$$X_{River}^{i+1} = X_{River}^{i} + rand \times C \times \left(X_{Sea}^{i} - X_{River}^{i}\right) \tag{11}$$

where *rand* is a uniformly distributed random number between 0 and 1. If the solution given by a stream is better than its connecting river, the positions of river and stream are exchanged (i.e. stream becomes river and river becomes stream). Such exchange can similarly happen for rivers and sea. Figure 3 depicts the exchange of a stream which is best solution among other streams and the river.

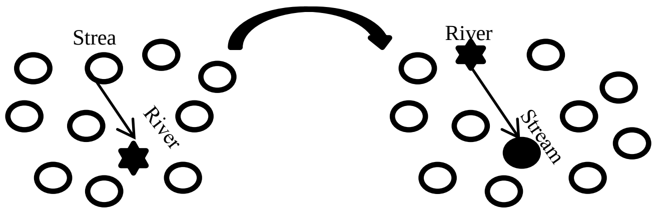

Figure 3. Exchange in positions of the river and the stream by Eskandar Et Al. [22].

Evaporation is one of the most important factors that can prevent the algorithm from rapid convergence (immature convergence). As can be seen in nature, water evaporates from rivers and lakes while plants give off (transpire) water during photosynthesis. The evaporated water is carried into the atmosphere to form clouds which then condenses in the colder atmosphere, releasing the water back to earth in the form of rain. The rain creates the new streams and the new streams flow to the rivers which flow to the sea.

This cycle which was mentioned in subsection 2.1 is called water cycle. In the proposed method, the evaporation process causes the sea water to evaporate as rivers/streams flow to the sea. This assumption is proposed in order to avoid getting trapped in local optima. The following Pseudocode shows how to determine whether or not river flows to the sea.

$$If \left|x_{Sea}^{i} - x_{River}^{i}\right| < d_{max} \quad i = 1,2,3, \dots, N_{sr} - 1 \tag{12}$$

where 'd_{max}'is a small number (close to zero). Therefore, if the distance between a river and sea is less than 'd_{max}', it indicates that the river has reached/joined the sea. In this situation, the evaporation process is applied and as seen in the nature after some adequate evaporation the raining (precipitation) will start. A large value for 'd_{max}' reduces the search while a small value encourages the search intensity near the sea. Therefore, 'd_{max}' controls the search intensity near the sea (the optimum solution). The value of 'd_{max}' adaptively decreases as:

$$d_{max}^{i+1} = d_{max}^{i} - \frac{d_{max}^{i}}{max \;\; iteration} \tag{13}$$

After satisfying the evaporation process, the raining process is applied. In the raining process, the new raindrops form streams in the different locations (acting similar to mutation operator in GA). For specifying the new locations of the newly formed streams, the following equation is used:

$$X_{Stream}^{new} = LB + rand \times (Ub - LB) \qquad (14)`$$

where LB and UB are lower and upper bounds defined by the given problem, respectively. Again, the best newly formed raindrop is considered as a river flowing to the sea. The rest of new raindrops are assumed to form new streams which flow to the rivers or may directly flow to the sea.

In order to enhance the convergence rate and computational performance of the algorithm for constrained problems, Eq. (15) is used only for the streams which directly flow to the sea. This equation aims to encourage the generation of streams which directly flow to the sea in order to improve the exploration near sea (the optimum solution) in the feasible region for constrained problems.

$$X_{Stream}^{new} = X_{Sea} + \sqrt{\mu} \times randn\,(1, N_{var}) \qquad 15)$$

where 'μ' is a coefficient which shows the range of searching region near the sea. '*randn*' is the normally distributed random number. The larger value for l increases the possibility to exit from feasible region. On the other hand, the smaller value for l leads the algorithm to search in smaller region near the sea. A suitable value for l is set to 0.1.

In mathematical point of view, the term $\sqrt{\mu}$ in Eq. (15) represents the standard deviation and, accordingly, 'μ' defines the concept of variance. Using these concepts, the generated individuals with variance 'μ' are distributed around the best obtained optimum point (sea).
For termination criteria, as commonly considered in metaheuristic algorithms, the best result is calculated where the termination condition may be assumed as the maximum number of iterations, CPU time, or e which is a small non-negative value and is defined as an allowable tolerance between the last two results. The WCA proceeds until the maximum number of iterations as a convergence criterion is satisfied.

Algorithm: Water Cycle Model ( )
The steps of WCA are summarized as follows:

```
Step 1:  Choose the initial parameters of the WCA: Nsr, dmax, Npop, max_iteration.
Step 2:  Generate random initial population and form the initial streams (raindrops), rivers, and sea using Eqs. (6) and (7).
Step 3:  Calculate the value (cost) of each raindrops using Eq. (5).
Step 4:  Determine the intensity of flow for rivers and sea using Eq. (8).
Step 5:  The streams flow to the rivers by Eq. (10).
Step 6:  The rivers flow to  sea which is the most downhill place using Eq. (11).
Step 7:  Exchange positions of river with a stream which gives the best solution, as shown in Fig. ab.
Step 8:  Similar to Step 7, if a river finds better solution than the sea, the position of river is exchanged with the sea (see Fig. ab).
Step 9:  Check the evaporation condition using the Pseudocode in subsection 2.2.3.
Step 10: If the evaporation condition is satisfied, the raining process will occur using Eqs. (14) and (15).
Step 11: Reduce the value of dmax which is user defined parameter using Eq. (13).
Step 12: Check the convergence criteria. If the stopping criterion is satisfied, the algorithm will be stopped, otherwise return to Step 5.
```

So at the end of this algorithm we'll have the information about performances of different subsets of attributes implemented on a common ANN over a set of similar data sets. More specifically we can say that from the attained information, we can precisely differentiate between the relevant set of attributes and irrelevant set of attributes and thereby we can simply ignore those irrelevant attributes while going for the estimating process. The proposed ANN used to differentiate the performances of several combinations of subset attributes is discussed in the next section.

Step 2: Proposed Artificial Neural Network Model
The performance of any Artificial Neural Network (ANN) is defined by its basic architecture which includes various parameters like number of hidden layers in ANN, the neuron (node) count in each of these layers, transfer function used at each node, weights and parameters of the training algorithm used including their settings too. The architecture we used in our study is discussed as:

In step1, 17 cost drivers are reduced to 11, these 11 drivers are then considered as input to the proposed ANN. The 11 drivers include size of the software project in KLOC, 09 effort multipliers, actual effort and one bias value. All these inputs are given as weighted inputs by associating a weight with each input and calculate the effort using the following Eq. 16:

$$\log(Effort) = \log(a \,*\, [\text{SIZE}]_b \text{ X } \textstyle\prod_{i=1}^{11} \text{EM}_i) \qquad (16)$$

Log of effort is needed as the received weighted inputs are summed up which otherwise in COCOMO are multiplied. To overcome this, log function is used to bring the neutrality. Moreover, weights are initialized to 1 and iterated through 1 to 11, learning rate 'μ' equals to 0.001 and bias 'b' equals to 1. So the above Eq.16 is a log transformed equation of COCOMO as given below in Eq.17.

$$\text{Effort} = a * [\text{size}]_b \text{ X } \prod_{i=1}^{15} \text{EM}_i \qquad (17)$$

The output obtained using equation 16, is compared using the activation function and the output signal is forwarded. Based on the value obtained out of the activation function, the weights applied on the inputs are tailored. Once the output from the activation function becomes 1, a difference of actual and calculated effort is considered and if found to be within permissible, the output is accepted otherwise the weights are adjusted again. This way a single full iteration of the project is completed called as epoch of the project. The algorithm for training and computing new set of weights for the above network is given as:

```
Step 1:     Initialize Weights, Learning rate 'μ'and Bias 'b'.
Step 2:     Repeat steps 03 to 08 until stopping condition is met.
```

Step 3: Repeat steps 04 to 07 for each pair of trainings.

Step 4: Pass signals to input layer, apply activation function on all input units from i = 1 to 11, and forward it to hidden layer.

Step 5: Each hidden unit j = 1 to 5 sums its weighted input to calculate the net input y_{inj} as per the 4:

$$y_{inJ} = b_j + \sum_{i=1}^{n} x_i w_{ij} \qquad (18)$$

The activation functions given below in Eq. 19 are applied to net input of Eq.18 to calculate the final output Yj as shown in Eq. 20:

$$f(y_n) = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if -\theta \le y_{in} \le \theta \\ -1 & if\ y_n < -\theta \end{cases} \qquad (19)$$

Where; θ represents threshold

$$Y_j = f(y_{inj}) \qquad (20)$$

Step 6: Calculate effort as the output given at output layer using the procedure given above in step 5 and also considering all the weights j=1 to 5 as 1.

Step 7: Make a comparison between actual effort and the computed effort. If the calculated difference is within the permissible limit, the current output is accepted otherwise the weights are updated using the equation 21.

$$w_i(new) = w_i(old) + \alpha \times input(i) \qquad (21)$$

Step 8: Test for stopping condition which means that if there is no change of weights then training process is stopped otherwise repeat from step3.

V. Datasets and Evaluation Criteria

A. *Datasets Used:*

For the purpose of assessing the SCE using the proposed model discussed above, only two data sets from different companies are chosen. One of the data sets has been got from the study of Mair et al. In this study, 32 data sets [37] were available publicly among which only one data set COCOMO 81 has been selected as this is the only one dataset containing data of more than 50 software development projects. In addition, one more public domain data set USP05 [38] became available after inspecting the recent literature. The reason of choosing only these two datasets is that the attribute count in both of these is restricted top 16 and 14 for COCOMO81 and USP05 datasets respectively.

B. *Evaluation Criteria*

Accuracy in SCE is evaluated based on the variation between the estimated effort and actual effort. A principle measure already used in the existing literature for these kinds of cost estimation models is Magnitude of Relative Error [39].

$$\text{MRE } (in\ \%) = \frac{Actual\ Effort - Estimated\ Effort}{Actual\ Effort}$$

We calculate MRE for each project. However, we can average the MRE of 'n' projects using their mean as:

$$\text{MMRE}_{(\text{Mean Magnitude of Relative Error})} = \frac{1}{n}\sum_{xi=1}^{n} \text{MRE}(xi)$$

Where; xi = 1 to 'n' projects.

MMRE is the commonly used evaluation criteria however being much exposed to outliers [40] we have used Median of Magnitude of Relative Error (MdMRE) as an evaluation criteria here instead of MRE. The MdMRE is given:

$$\text{MdMRE} = 100 \times Median\ (\text{MRE})$$

VI. Experimentation Results

After implementing the proposed procedure, here in this section, the MRE value of proposed model is calculated for randomly selected set of 10 projects from both the datasets and then compared with the results obtained using COCOMOII model which is considered to be the basic estimation model in software development projects cost estimation and with another existing model by [41]. Below two tables namely Table 2 and Table 3 showed the substantial difference in MRE of the proposed technique against the two existing techniques, followed by Figure 4 and Figure 5illustrating the graphical demonstration of all of the three models (COCOMO II, B.T Rao, et al. Model and Proposed Model) when executed on COCOMO 81 and USP05datasets respectively.

Table 2. MRE (%) of proposed model and other two existing

S No	Project ID	MRE (%) on USP05		
		COCOMO-II Model	B.T Rao, et al. Model	Proposed Model
01	1	9.33	5.12	3.60
02	5	8.84	3.78	2.70
03	15	16.75	8.80	6.88
04	25	14.09	5.68	3.90
05	30	8.81	4.40	3.04
06	42	13.9	6.60	4.28
07	54	13.67	7.80	4.76
08	60	11.78	6.83	5.76
09	62	13.2	5.43	2.20
10	75	17.09	8.66	4.76

techniques on 10 randomly selected projects of COCOMO81dataset.

Table 3. MRE (%) of proposed model and other two existing

S No	Project ID	MRE (%) on COCOMO dataset		
		COCOMO-II Model	**B.T Rao, *et al.* Model**	**Proposed Model**
01	05	7.44	5.23	3.26
02	12	19.83	9.18	6.98
03	30	6.49	3.21	2.12
04	38	50.98	14.40	11.18
05	40	12.40	4.89	3.79
06	45	5.35	3.69	2.90
07	47	16.40	7.60	4.02
08	59	8.66	3.20	2.96
09	61	13.10	8.61	3.98
10	62	6.22	3.30	2.80

techniques on 10 randomly selected projects of USP05 dataset

As Mean of Magnitude of Relative Error (MMRE) is exposed to outliers, we use Median of Magnitude of Relative Error (MdMRE) as our second evaluation criteria; the MdMRE of all of the three techniques was calculated on both of the datasets and given in Table 4 followed by graphical depiction to be given in Figure 6 respectively. From this figure it is again clear that the proposed models outperforms well.

Table 4. MdMRE of proposed model and other two existing techniques on 10 randomly selected projects of COCOMO and USP05 Datasets

	MdMRE Comparison		
	COCOMO II Model	B.T Rao, *et al.* Model	Proposed Model
COCOMO Dataset (10 Projects of Table 1)	10.53	5.06	3.35
USP05 Dataset (10 Projects of Table 2)	13.43	6.14	2.66

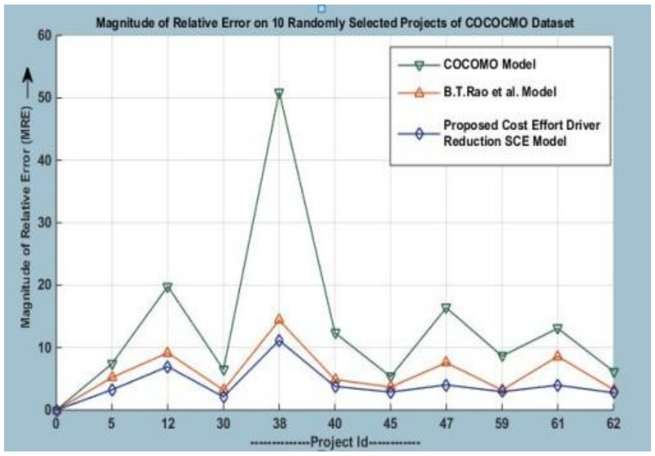

Figure 4. MRE (%) based graphical description of two existing models and proposed model on COCOMO data set

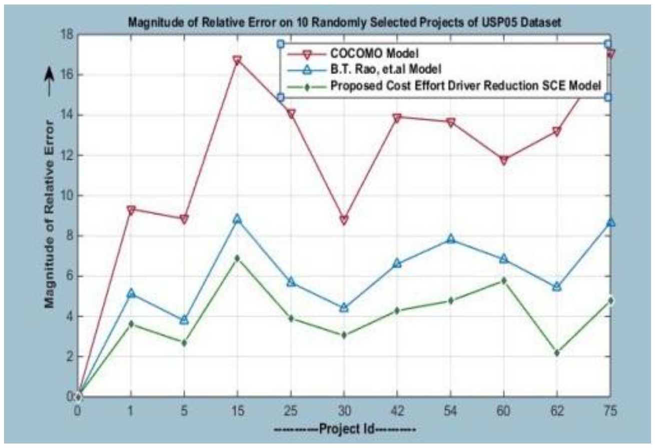

Figure 5. MRE (%) based graphical description of two existing models and proposed model on USP05data set

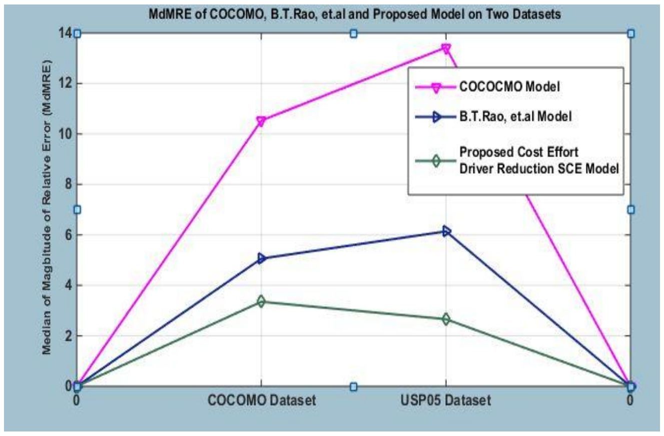

Figure 6. MdMRE (%) based graphical description of two existing models and proposed model on COCOMO and USP05 data sets

VII. CONCLUSION AND FUTURE SCOPE

This research study concludes that estimating software development cost by introducing the above two step process yields higher performance as the inclusion of cost driver reduction procedure results in a model which is more stable as budding collinearity between attributes is minimized. Moreover, a model with lesser number of attributes is usually chosen over more attributed models as it always becomes easier to interpret such models. Moreover, the proposed model in addition to attainment of accurate software cost estimation results, also trims the computational complexity without any forfeit on the performance.

In future however, it is possible to combine the proposed cost driver reduction procedure with some other high order artificial neural networks like Functional Link ANN to get even more positive results up in the domain of software cost estimation.

REFERENCES

[1] Auer, Martin, et al. "*Optimal project feature weights in analogy-based cost estimation*: Improvement and limitations." IEEE Transactions on Software Engineering 32.2 (2006): 83-92.

[2] Baskeles, Bilge, Burak Turhan, and Ayse Bener. "*Software effort estimation using machine learning methods*" Computer and information sciences, 2007. iscis 2007. 22nd international symposium on. IEEE, 2007.

[3] Boehm, Barry W. "*Software engineering economics*" Vol. 197. Englewood Cliffs (NJ): Prentice-hall, 1981.

[4] Heemstra, Fred J. "*Software cost estimation*" Information and software technology 34.10 (1992): 627-639.

[5] Huang, Sun-Jen, and Nan-Hsing Chiu. "*Optimization of analogy weights by genetic algorithm for software effort estimation*" Information and software technology 48.11 (2006): 1034-1045.

[6] Li, Jingzhou, and Guenther Ruhe. "*A comparative study of attribute weighting heuristics for effort estimation by analogy*" Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering. ACM, 2006.

[7] Li, Jingzhou, et al. "*A flexible method for software effort estimation by analogy*" Empirical Software Engineering 12.1 (2007): 65-106.

[8] Li, Jingzhou, and Guenther Ruhe. "*Decision support analysis for software effort estimation by analogy*" Proceedings of the Third International Workshop on Predictor Models in Software Engineering. IEEE Computer Society, 2007.

[9] Shepperd, Martin, and Chris Schofield. "*Estimating software project effort using analogies*" IEEE Transactions on software engineering 23.11 (1997): 736-743.

[10] Chiu, Nan-Hsing, and Sun-Jen Huang. "*The adjusted analogy-based software effort estimation based on similarity distances*" Journal of Systems and Software 80.4 (2007): 628-640.

[11] Costagliola, Gennaro, et al. "*Effort estimation modeling techniques: a case study for web applications*" Proceedings of the 6th international conference on Web engineering. ACM, 2006.

[12] Jørgensen, Magne. "*A review of studies on expert estimation of software development effort*" Journal of Systems and Software 70.1-2 (2004): 37-60.

[13] Menzies, Tim, et al. "*Selecting best practices for effort estimation*" IEEE Transactions on Software Engineering 32.11 (2006): 883-895.

[14] Rush, Christopher, and Rajkumar Roy. "*Expert judgement in cost estimating: Modelling the reasoning process*" Concurrent Engineering 9.4 (2001): 271-284.

[15] Mair, Carolyn, and Martin Shepperd. "*The consistency of empirical comparisons of regression and analogy-based software project cost prediction*" Empirical Software Engineering, 2005. 2005 International Symposium on. IEEE, 2005.

[16] Chen, Zhihao, et al. "*Finding the right data for software cost modeling*" IEEE software 22.6 (2005): 38-46.

[17] Zhang, Mi, and J. T. Yao. "*A rough sets based approach to feature selection*" Fuzzy Information, 2004. Processing NAFIPS'04. IEEE Annual Meeting of the. Vol. 1. IEEE, 2004.

[18] Turhan, Burak, Onur Kutlubay, and Ayse Bener. "*Evaluation of feature extraction methods on software cost estimation*" Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on. IEEE, 2007.

[19] Mendes, Emilia, et al. "*A comparative study of cost estimation models for web hypermedia applications*" Empirical Software Engineering 8.2 (2003): 163-196.

[20] Azzeh, Mohammad, Daniel Neagu, and Peter Cowling. "*Improving analogy software effort estimation using fuzzy feature subset selection algorithm*" Proceedings of the 4th international workshop on Predictor models in software engineering. ACM, 2008.

[21] Mittal, Mamta, et al. "*Monitoring the Impact of Economic Crisis on Crime in India Using Machine Learning*" Computational Economics (2018): 1-19.

[22] Venkatachalam, A. R. "*Software cost estimation using artificial neural networks*" Neural Networks, 1993. IJCNN'93-Nagoya. Proceedings of 1993 International Joint Conference on. Vol. 1. IEEE, 1993.

[23] Mamta Mittal, Lalit Mohan Goyal, Sumit Kaur, Iqbaldeep Kaur, Amit Verma, D. Jude Hemanth, "*Performance Enhanced Growing Convolutional Neural Network Based Approach for Brain Tumor Segmentation in Magnetic Resonance Brain Images*", Applied Soft Computing.

[24] Gray, Andrew R. "*A simulation-based comparison of empirical modeling techniques for software metric models of development effort*" Neural Information Processing, 1999. Proceedings. ICONIP'99. 6th International Conference on. Vol. 2. IEEE, 1999.

[25] Malav Shastri, Sudipta Roy, Mamta Mittal, (2019) "*Stock Price Prediction using Artificial Neural Model : An Application of Big Data*", SIS, EAI, (ESCI Indexed) DOI: 10.4108/eai.19-12-2018.156085.

[26] Xu, Zhiwei, and Taghi M. Khoshgoftaar. "*Identification of fuzzy models of software cost estimation*" Fuzzy Sets and Systems145.1 (2004): 141-163.

[27] Hughes, R. T. "*An evaluation of machine learning techniques for software effort estimation*" University of Brighton (1996): 1-15.

[28] Jorgensen, Magne. "*Experience with the accuracy of software maintenance task effort prediction models*" IEEE Transactions on software engineering 21.8 (1995): 674-681.

[29] Samson, Bill, David Ellison, and Pat Dugard. "*Software cost estimation using an Albus perceptron (CMAC)*" Information and Software Technology 39.1 (1997): 55-60.

[30] Heiat, Abbas. "*Comparison of artificial neural network and regression models for estimating software development effort*" Information and software Technology 44.15 (2002): 911-922.

[31] Serluca, C. "*An investigation into software effort estimation using a back propagation neural network*" Mémoire de maîtrise, Bournemouth University, Grande-Bretagne (1995).

[32] Srinivasan, Krishnamoorthy, and Douglas Fisher. "*Machine learning approaches to estimating software development*

effort" IEEE Transactions on Software Engineering 21.2 (1995): 126-137.
[33] Wittig, Gerhard, and Gavin Finnie. "*Estimating software development effort with connectionist models*" Information and Software Technology 39.7 (1997): 469-476.
[34] Schofield, Chris. "*Non-algorithmic effort estimation techniques*" ESERG, TR98-01 (1998).
[35] Eskandar, Hadi, et al. "*Water cycle algorithm–A novel metaheuristic optimization method for solving constrained engineering optimization problems*" Computers & Structures110 (2012): 151-166.
[36] David, S. "*The water cycle, illustrations by John Yates*" New York: Thomson Learning (1993).
[37] Mair, Carolyn, Martin Shepperd, and Magne Jørgensen. "*An analysis of data sets used to train and validate cost prediction systems*" ACM SIGSOFT software engineering notes. Vol. 30. No. 4. ACM, 2005.
[38] Menzies, Tim, et al. "*Selecting best practices for effort estimation*" IEEE Transactions on Software Engineering 32.11 (2006): 883-895.
[39] S.D. Conte, H.E. Dunsmore, and V.Y. Shen, "*Software Engineering Metrics and Models*". The Benjamin/Cummings Publishing Company, Inc., 1986.
[40] Port, Dan, and Marcel Korte. "*Comparative studies of the model evaluation criterions mmre and pred in software cost estimation research*" Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement. ACM, 2008.
[41] Rao, B. Tirimula, et al. "*A novel neural network approach for software cost estimation using Functional Link Artificial Neural Network (FLANN)*" International Journal of Computer Science and Network Security 9.6 (2009): 126-131.

Authors Profile

Dr. Zahid Hussain Wani is working as an Assistant Professor in Department of Computer Sciences, Islamic University of Science & Technology, J & K, He has received his Ph.D. in Computer Science from the Department of Computer Sciences, University of Kashmir, India. He has received his bachelor's degree in Computer Applications from University of Kashmir and his master's degree in Computer Applications from Islamic University of Science & Technology. His research interests include Artificial Neural Networks, Data Mining, Machine Learning, Big Data Analytics and Software Engineering. He has several journal papers, some conference papers and few best paper awards to his credit too.

Dr. Rumaan Bashir is working as a Senior Assistant Professor and currently heading the Department of Computer Sciences, Islamic University of Science & Technology, J & K, She has received her Ph.D. in Computer Science, her MPhil, and her Master Degree in Computer applications from the Department of Computer Sciences, University of Kashmir, India. Her research interests include Digital Image Processing, Natural Language Processing, Information Security, Machine Learning, Big Data Analytics and Software Engineering. To her achievements, she has got a Gold medallist in her Post Grad and had been selected as Assistant Professor at 10+2+3in Jammu & Kashmir Public Service Commission and stood at rank 1st both in written as well as final Selection List. She has a good number of journal papers of very high repute, a good number of conference papers and few best paper awards to her credit too.

Dr. Kaiser J. Giri: Has received his Master's Degree, M.Phil & Ph. D. in Computer Applications from University of Kashmir, India, The Author is working as an Sr. Assistant Professor in the Department of Computer Science and Co-ordinator at Advanced Center for Information Technology & Gov., Islamic University of Science & Technology, and Awantipora, J & K, India. The interest arrears of author include Image Processing, Signal Processing, and Information Security, Natural Language Processing, Machine Learning, Software Engineering. He has a good number of journal papers of very high repute, and also a good number of conference papers to his credit too.